Remarks

Status of application

Claims 1-60 were examined and stand rejected in view of prior art, as well as stand rejected for technical issues. The claims have been amended to further clarify Applicant's invention and to address objections raised by the Examiner. Reexamination and reconsideration are respectfully requested.

The invention

Applicant's invention comprises a system providing methods for securing the internal interfaces of executable components of a program, so as to better guard against malicious attacks seeking to misuse these internal interfaces so as to attack the program and/or the computing system(s) on which the program is used. In a normal application environment, a program typically consists of a number of interoperable components. For example, an executable file such as a dynamic link library (DLL) or executable file (.exe) frequently "exports" certain functions to make them available to other program components and includes information about these exported functions in export tables of the program. Unfortunately, information in conventional export tables can be obtained and exploited by those seeking to attack the program or the system(s) on which the program is used.

Applicant's methodology provides for post-processing export tables of executable components that have been built in a conventional fashion. The post-processing of an export table includes reformatting the table, extracting information from the table, (optionally) shrinking what remains of the table (i.e., reducing its size in the file), and then placing the extracted information in a protected table so as to better secure the program against attack. As part of the post-processing of an export table, information regarding any export (i.e., exported function) in the export table that is to be secured is entirely removed from the original export table. This makes it more difficult for an attacker to locate and call these functions as a means of attack. This is an improvement over prior solutions which typically obscure function names, but do not remove them from the export table.

Applicant's invention makes it much more difficult to identify an avenue for attacking the program by removing function names from the export table, as well as hiding the addresses of these functions. The removal of clear text function names from the original export table makes it more difficult for an attacker to determine a function that can be used to attack a program. Removing information from the export table also makes it more difficult for an attacker to find the address of the function in order to initiate an attack. Applicant's methodology makes it more difficult for an attacker to use certain well-documented system calls to determine the address of a function that is targeted for attack.

Applicant's methodology also provides additional security through forward and backward validation of the code signature of program modules. After loading an executable file, but before calling any of its secured exports, both the code signatures of the importer and the exporter are validated. Access to the exported function is only provided if the code signatures are validated. In this manner, attempts to invoke the exported function by components that are not authenticated (e.g., attempts to misuse the exported function to attack the program or the system(s) on which it is running) are blocked.

General

A. Section 101 rejection

Claims 1-12, 24-28, 30-58 and 60 stand rejected under 35 U.S.C. 101 on the basis of non-statutory subject matter. Here, the Examiner states that Applicant's claimed invention is directed to a practical application; however, the limitations fail to produce a useful, concrete, and tangible result. Applicant has amended independent claims 1 and 45 to include claim limitations of securing program by blocking attempts to invoke an exported function of a program if the component attempting to invoke the exported function is not authenticated. Applicant's independent claims 15 and 30 were not amended as these claims already include claim limitations of blocking access by components that are not authenticated, so as to secure against unauthorized access. Accordingly, it is respectfully submitted that in view of the above-mentioned amendments, the rejection of Applicant's claims 1-12, 24-28, 30-58 and 60 under Section 101 on the basis of non-statutory subject matter is overcome.

Prior art rejections

A. Section 102(b): Ferguson

Claims 1-10, 12-19, 21-22, 24-37, 39-42, 45, 48-54, and 56-60 stand rejected under 35 U.S.C. 102(b) as being anticipated by Ferguson, U.S. Pat. 5,933,826 ("Ferguson"). The Examiner's rejection of Applicant's claim 1 as follows is representative of the Examiner's rejection of Applicant's claims as anticipated by Ferguson:

> (1) with regard to claim 1:
> A method for securing a program comprised of a plurality of interoperable components, the method comprising:
> extracting information about a function [step 83, Col 8, lines 33-34] of a first component of the program that is callable by at least one other component of the program ['interpreter'; Col 8, lines 35-36];
> securing the extracted information [Col 8, lines 38-39]; in response to an attempt by a second component of the program to invoke the function of the first component, validating authenticity of the second component [Col 8, lines 45-48]; and
> if the second component is validated, providing access to the function of the first component using the secured extracted information [Col 8, lines 49-52].

Under Section 102, a claim is anticipated only if each and every element as set forth in the claim is found, either expressly or inherently described, in the single prior art reference. (See, e.g., MPEP Section 2131.) As will be shown below, Ferguson fails to teach each and every element set forth in Applicant's claims 1-10, 12-19, 21-22, 24-37, 39-42, 45, 48-54, and 56-60 (as well as other claims), and therefore fails to establish anticipation of the claimed invention under Section 102.

Ferguson describes an approach for securing objects stored in a distributed directory. Ferguson's system comprises a plurality of interconnected nodes which access a distributed directory including a hierarchy of objects, each of which have associated attributes. The distributed directory is a synchronized hierarchical database which maintains and provides access to information across the network of interconnected nodes (Ferguson, col. 4, lines 19-25). Information on the distributed directory can be created, read, modified, and shared by nodes having access rights to the distributed directory (Ferguson, col. 4, lines 33-36). Access to objects in the distributed directory of Ferguson is controlled using an access control mechanism which provides physical security, login security and directory security for the distributed directory (Ferguson, col. 6, lines 46-49, Fig. 3). The physical security and login security measures described by Ferguson are conventional steps of maintaining physical security of computer systems and requiring a user name and password for login to such systems (Ferguson, col. 6, lines 55-64). Ferguson also describes that directory security which is used after login security has been first verified to assign specified rights to the objects stored in the distributed directory, such as rights to read, write, create and erase such objects (Ferguson, col. 7, lines 17-30) and rights to access or manage objects and/or change their properties (Ferguson col. 7, lines 33-44). Although Ferguson describes mechanisms for securing objects in a distributed directory, it is distinguishable from Applicant's claimed invention in a number of respects.

One initial difference is that Applicant's system is directed to securing internal interfaces of components of a program comprised of plurality of interoperable components. Applicant's invention addresses a specific security vulnerability of these programs in that a perpetrator seeking to attack a program can use export information (e.g., in export tables of a component) to identify a function that can be exploited in an attack and to determine the location of the function (Applicant's specification, paragraph [0062]). Applicant's invention provides for securing this export information, so as to block attacks which attempt to obtain the function name, address or other information in order for use in an attack on the program (Applicant's specification, paragraph [0067]). In this manner, Applicant's invention operates at runtime when an exported function of a program is invoked to only allow invocation of such function if the authenticity of the importer is validated. Applicant's review of Ferguson finds no comparable teaching of securing export information or other internal interfaces of a program consisting of interoperable components so as to guard against attacks on internal interfaces of a program as such program is in operation.

Another significant difference from Ferguson is that Applicant's approach provides for securing export information about an exported function of a program component by extracting (i.e., removing) this export information from an export table of the component (Applicant's specification, paragraphs [0066] - [0067]). These features are specifically included in Applicant's claims. For example, Applicant's claim 1, as amended, includes the following claim limitations:

> A method for <u>securing a program comprised of a plurality of interoperable components</u>, the method comprising:
> <u>extracting export information about a function of a first component of the program that is callable by at least one other component</u> of the program;
> securing the extracted export information;
>
> (Applicant's claim 1, as amended, emphasis added)

As shown, Applicant's methodology provides for extracting the export information from the export table. The extracted information is then secured so as to make it more difficult for an attacker to determine the name or address of a function in order to use such information as a basis for attacking the program (Applicant's specification, paragraph [0067]).

Even assuming the distributed directory of Ferguson is somehow analogous to the export tables described by Applicant (which Applicant does not believe is correct), <u>Ferguson provides no teaching of extracting information from the distributed directory so as to secure such information</u>. The Examiner cites Ferguson at col. 8, lines 33-36 for the teaching of extracting information about an exported function of a program component; however, the referenced portion of Ferguson provides as follows:

> Continuing to step 83, a program, such as a script, pseudo code or object code, is accessed. In the case of script or pseudo code, an interpreter to execute the program may optionally be accessed as well.
> A further level of security can be achieved in optional step 84, wherein the program is encrypted using an encryption system.
>
> (Ferguson, col. 8, lines 33-36)

As illustrated above, Ferguson describes (optionally) encrypting a program so as to secure the program, but provides no teaching of extracting information about exported functions of the program in the manner described in Applicant's specification and claims. In contrast to Applicant's approach which provides for <u>removing information</u> (export information) from export tables so as to better secure such information, Ferguson's system stores items in the distributed directory and then controls access to the items stored within the distributed directory. This is, for example, described by Ferguson as follows:

> Using the present invention, programs can be stored and transmitted in a distributed directory, and utilize the powerful and flexible security mechanisms provided with the distributed directory. Consider the flow chart depicted in FIG. 4. In step 81, after satisfying the applicable access requirements a distributed directory is accessed. The distributed directory has a hierarchy of objects with associated attributes. The distributed directory also has an access control mechanism, such as physical, login and/or directory security, for controlling access to at least a portion of the distributed directory.
>
> (Ferguson, col. 8, lines 19-29, emphasis added)

Applicant's approach of removing the export information from the export tables provides additional security as it makes it more difficult for an attacker to determine the function name and the address of the function for purposes of initiating an attack using this export information. Removing and securing the export information provides for improved security as an attacker cannot use certain well-known system tools and utilities to obtain the address of the exported function (Applicant's specification, paragraphs [0078] and [0098]). Applicant's invention regulates access to the secured export information so as to only permit properly authenticated components of the program to use such information, while denying access and use of the secured export information by unauthorized components.

As an executable program file is loaded, the code signatures of both importer and exporter components of the program are validated (Applicant's specification, paragraph [0069]). In operation, when another component (e.g., an importer component of the program) attempts to invoke the exported function, the authenticity of the importer component is validated before access and use of the extracted export information for purposes of invoking the function is allowed (Applicant's specification, paragraph [0074]). The exported function may only invoke the function using the extracted export information if the authenticity of the importer is validated (Applicant's specification, paragraph [0075]). If the authenticity of the importer is not validated, the importer's attempt to invoke the exported function fails (Applicant's specification, paragraph [0073]). These features are also described in Applicant's claims including for example, in the following limitations of claim 1:

> in response to an attempt by a second component to invoke the function of the first component, validating authenticity of the second component;
> <u>if the authenticity of the second component is validated, providing access to the function of the first component using the secured extracted export information</u>; and
> otherwise, blocking the attempt by the second component to invoke the function.
>
> (Applicant's claim 1, as amended, emphasis added)

Although Ferguson mentions authentication, it does so in the context of remote access to the distributed directory by outside parties. In this regard, Ferguson discusses user authentication (e.g., based on password login). However, Ferguson does not include the specific teaching of Applicant's claims of securing export information and only permitting another program component (e.g., an importer) to access and use the secured export information for invoking an exported function during runtime operations if the authenticity of the component is validated.

All told, Ferguson provides no teaching of securing internal interfaces of a program consisting of multiple interoperable components by extracting and securing export information about exported functions and only permitting access and use of such export information for by authenticated importer components in the manner described in Applicant's claims. Therefore, as Ferguson does not teach or suggest all of the claim limitations of Applicant's claims 1-10, 12-19, 21-22, 24-37, 39-42, 45, 48-54, and 56-60 (and other claims) it is respectfully submitted that the claims distinguish over this reference and overcome any rejection under Section 102.

B. Section 103(a): Ferguson in view of Idoni

Claims 11, 20, 23, 38, 43-44, 46-47, and 55 stand rejected under 35 U.S.C. 103(a) as being unpatentable over Ferguson (above), in view of Idoni, US Published Application 2004/0123308 ("Idoni"). The Examiner's rejection of claims 11, 23, 38 and 55 as follows is representative of the Examiner's rejection of Applicant's claims as being unpatentable over Ferguson in view of Idoni:

> with regard to claims 11, 23, 38, and 55:

> Ferguson teaches a method for securing a program comprised of a plurality of interoperable components, comprising: extracting information about a function [Col 8, lines 33-34] of a first component of the program that is callable by at least one other component of the program [Col 8, lines 35-36]; a method of transparent data delivery comprising: securing the extracted information [Col 8, lines 38-39]; in response to an attempt by a second component of the program to invoke the function of the first component, validating authenticity of the second component [Col 8, lines 45- 48]; and if the second component is validated, providing access to the function of the first component using the secured extracted information [Col 8, lines 49-52].
> Ferguson does not teach returning an address of the function as required in claims 11, 23, 38, and 55.
> However, Idoni teaches a method that provides a step that includes returning an address of the function if authenticated [Col 3, paragraph 0031]. The 'physical location' disclosed by Idoni is interpreted by the Examiner as the return address of the function or module. Furthermore, the method taught by Idoni provides the advantage of both implicit and explicit linking while eliminating much of the programmatic and associated overhead at design, implementation and execution time [Col 2, paragraph 0017].
> Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to recognize that the identification and physical location of a function used in the method taught by Idoni would have been implemented in the method disclosed by Ferguson in order to identify the physical location of the dynamic link loader [Col 3, lines 16-19]. Access can then be regulated to preserve security.

Under Section 103(a), a patent may not be obtained if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which the subject matter pertains. To establish a prima facie case of obviousness under this section, the Examiner must establish: (1) that there is some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings, (2) that there is a reasonable expectation of success, and (3) that the prior art reference (or references when combined) must teach or suggest all the claim limitations. (See e.g., MPEP 2142). As will be shown below, the Ferguson and Idoni references, even when combined, fail to meet the requisite condition of teaching or suggesting all of Applicant's claim limitations.

Initially, the claims are believed to be allowable for at least the reasons cited above (as to the Section 102 rejection) pertaining to the deficiencies of Ferguson as to

Applicant's invention. Idoni does not cure these deficiencies of Ferguson as it includes no teaching of extracting export information about exported functions of a program and securing the extracted export information as provided in Applicant's specification and claims. Moreover, Applicant's review of Idoni finds that it does not, in fact, include the specific teachings referenced by the Examiner. The Examiner cites paragraph [0031] of Idoni for the teaching of a method that "provides a step that includes returning an address of the function <u>if authenticated</u>". However, review of paragraph [0031] and the balance of Idoni <u>finds no reference whatsoever to authentication</u> of the importer (or any other authentication) before the address of the exported function (e.g., DLL) is provided to the importer. Instead, Idoni states that the importer provides the physical location of the exported function as follows:

> In a step 504, the application program is loaded and executed by the computer operating system as defined for explicit linkage, including the unresolved references provided by the import libraries specified in step 502. In a step 506, the executing application invokes the DLL loader routine for any necessary DLLs. <u>Included in the call to the loader routine is an identification and physical location of the DLL</u>. In a step 508, the executing application invokes the linker routine to resolve any external references. <u>Included in the call to the linker program is a memory base address of the loaded DLL necessary to resolve external references</u>.
>
> (Idoni, col. 3, paragraph [0031], emphasis added)

As illustrated above, in Idoni's system the identification and physical location of the DLL is included in the call to the DLL loader routine. Presumably, this information is stored in (and obtained from) the program's import libraries as referenced above at step 502 of Idoni's method. This is not comparable to Applicant's approach. Applicant's approach provides for routing calls to an exported function that has been secured through a local "stub" function that has been inserted during the loading for the importer instead of looking up the address of the exported function in the import directory (Applicant's specification, paragraph [0075]). Significantly, this "stub" function is only inserted if the importer is authenticated (i.e., identified as a trusted program). This stub function routes the call to a security module which, in turn, locates the exported function that is being called, thereby enabling an authenticated importer to obtain the address of the exported function (Applicant's specification, paragraphs [0100]-[0101]). However, note that if the importer is not authenticated, these steps do not occur and the importer will not be able to obtain the address of the exported function.

As discussed above, neither Ferguson nor Idoni includes the teaching of extracting export information of an exported function of a program and securing this export information as provided in Applicant's claims. In addition, Idoni does not include the specific teachings of Applicant's claims of providing information about the exported function (e.g., its address) to an importer only if the authenticity of the importer is validated. Accordingly, as the prior art reference(s), even when combined, fail to teach or suggest all the claim limitations, it is respectfully submitted that Applicant's claimed invention as set forth by these claims 11, 20, 23, 38, 43-44, 46-47, and 55 is distinguishable over the two references, and that the rejection under Section 103 is overcome.

Any dependent claims not explicitly discussed are believed to be allowable by virtue of dependency from Applicant's independent claims, as discussed in detail above.

Conclusion

In view of the foregoing remarks and the amendment to the claims, it is believed that all claims are now in condition for allowance. Hence, it is respectfully requested that the application be passed to issue at an early date.

If for any reason the Examiner feels that a telephone conference would in any way expedite prosecution of the subject application, the Examiner is invited to telephone the undersigned at 408 884 1507.

Respectfully submitted,

Date: February 2, 2007

/John A. Smart/

John A. Smart; Reg. No. 34,929
Attorney of Record

408 884 1507
815 572 8299 FAX